Simulation Modelling Practice and Theory 37 (2013) 1–17

Contents lists available at SciVerse ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Hybrid system modeling using the SIMANLib and ARENALib Modelica libraries ☆

Victorino Sanz [a,*], Alfonso Urquia [a], François E. Cellier [b], Sebastian Dormido [a]

[a] Dpto. Informática y Automática, UNED, Juan del Rosal 16, 28040 Madrid, Spain
[b] Dept. Computer Science, ETH Zurich, CH-8092 Zurich, Switzerland

ARTICLE INFO

Article history:
Received 20 June 2012
Received in revised form 4 May 2013
Accepted 6 May 2013
Available online 14 June 2013

Keywords:
Modelica
Object-oriented modeling
Hybrid systems
Arena
SIMAN
Parallel DEVS

ABSTRACT

The ARENALib and SIMANLib Modelica libraries replicate the basic functionality of the Arena simulation environment and the SIMAN language. These libraries facilitate describing discrete-event models using the Arena modeling methodology. ARENALib and SIMANLib models can be combined with other Modelica models in order to describe complex hybrid systems (i.e., combined continuous-time and discrete-event systems). The implementation and design of SIMANLib and ARENALib is discussed. The ARENALib components have been built in a modular fashion using SIMANLib. The SIMANLib components have been described as Parallel DEVS models and implemented using DEVSLib, a Modelica library previously developed by the authors to support the Parallel DEVS formalism. The use of Parallel DEVS as underlying mathematical formalism has facilitated the development and maintenance of SIMANLib. The modeling of two hybrid systems is discussed to illustrate the features and use of SIMANLib and ARENALib: firstly, a soaking-pit furnace; secondly, the malaria spread and an emergency hospital. DEVSLib, SIMANLib and ARENALib can be freely downloaded from http://www.euclides.dia.uned.es/.

1. Introduction

The Arena simulation environment uses a flowchart-based modeling methodology that facilitates the description of discrete-event systems [8,9]. Systems are described using Arena from the point of view of the entities that flow through them using the available resources. Arena models are structured in a hierarchical and modular way. They are defined by means of a flowchart diagram and static data.

The flowchart diagram is composed by instantiating and connecting predefined components named flowchart modules. Each flowchart module has an interface and an internal behavior. The interface is used to connect with other modules, thus describing the path for the entities. The internal behavior describes the actions performed by the entities while in the component, e.g., delay a certain amount of time, seize and release resources, and record statistics. The simulation results are usually presented in the form of statistical indicators that are calculated during the simulation.

The static data allow to specify component characteristics, such as for instance, the characteristics of the entity arrival processes, resources and queues. The static data are described using the data modules provided to this end by Arena.

The Arena flowchart and data modules are arranged into panels. The main Arena panel is named BasicProcess. This panel includes the Create, Dispose, Process, Decide, Batch, Separate, Assign and Record flowchart modules, and the Entity, Queue,

☆ This work has been supported by the Spanish CICYT under DPI2007-61068 Grant.
* Corresponding author. Tel.: +34 91 3989469.
E-mail addresses: vsanz@dia.uned.es (V. Sanz), aurquia@dia.uned.es (A. Urquia), francois.cellier@inf.ethz.ch (F.E. Cellier), sdormido@dia.uned.es (S. Dormido).

1569-190X/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.simpat.2013.05.005

Resource, Variable, Schedule and Set data modules [8]. Other Arena panels are AdvanceProcess, AdvanceTransfer, AgentUtil, FlowProcess and Packaging [8].

Arena is based on the SIMAN simulation language [15]. Arena modules are high-level constructs whose functionality is equivalent to sets of SIMAN blocks and elements. Arena predefined modules are internally built using SIMAN blocks and elements, which represent lower-level actions. SIMAN and Arena components can be combined in the same model, provided that they have compatible interfaces and manage the same types of information. Arena provides two panels, named Blocks and Elements, that correspond to the components of the SIMAN language.

Arena provides limited support to the description of continuous-time models [8]. The linear models whose state-variable derivatives remain constant between discrete events can be described using the Levels and Rates SIMAN blocks, following the System Dynamics approach [7]. A general-purpose programming language (e.g., C, FORTRAN and Visual Basic) needs to be used to describe other types of continuous-time models and their connection to the Arena discrete-event model.

Other tools support different approaches to hybrid system modeling. For instance, Anylogic facilitates describing continuous-time models using System Dynamics [7], similarly to Arena. Ptolemy II provides functionality to describe the continuous-time model using block diagrams, similarly to Matlab/Simulink. Based on Ptolemy II, the Building Controls Virtual Test Bed (BCVTB) is a software environment that allows the co-simulation between different simulation programs, including the Modelica modeling environment Dymola among others [20].

The general-purpose, object-oriented modeling languages support the physical modeling paradigm [1]. In particular, the Modelica language [11] facilitates the object-oriented description of DAE-hybrid models, i.e., models composed of differential and algebraic equations, and discrete-time events. Modelica supports a declarative description of the continuous-time part of the model (i.e., equation-oriented modeling) and provides language expressions for describing discrete-time events. These features have facilitated the development of Modelica libraries supporting several modeling formalisms and describing phenomena in different physical domains [12]. Modelica facilitates the reuse of models and model components, which contribute to reduce the cost of new model development [17].

A number of Modelica libraries have been developed for supporting discrete-event modeling formalisms, including StateCharts [6], state graphs [14], hybrid automata [16], Petri Nets [13], extended Petri Nets [5] and Parallel DEVS [19]. The description of operation management models using Modelica is analyzed in [10] and the model of an inventory system is presented.

The Arena modeling methodology is supported by the SIMANLib and ARENALib Modelica libraries [18]. These libraries facilitate the description of discrete-event logistic models in Modelica, such as manufacturing and packaging processes, supply chains, health care systems, and transport and distribution networks among others. The physical modeling paradigm supported by Modelica can be combined with the Arena modeling methodology in order to describe complex hybrid systems. This combination is not currently supported by other modeling and simulation environments. SIMANLib and ARENALib include components to interface with other Modelica models. These components facilitate connecting models composed using SIMANLib and ARENALib to Modelica models developed using other methodologies, e.g., continuous-time models described using the physical modeling paradigm.

The implementation of SIMANLib and ARENALib, and their use for hybrid system modeling are discussed in this manuscript. Remarks on the SIMANLib and ARENALib implementation are provided in Section 2. The architecture and most relevant features of SIMANLib and ARENALib are described in Sections 3 and 4, respectively. The SIMANLib and ARENALib components for interfacing with models developed using other Modelica libraries are described in Section 5. Finally, two case studies are used to illustrate the SIMANLib and ARENALib capabilities for hybrid system modeling. The modeling of a soaking pit furnace and an emergency hospital are discussed in Sections 6 and 7, respectively. These case studies illustrate the capabilities for hybrid system modeling included in the libraries (i.e., external processes and generation of entities on demand).

2. Remarks on the SIMANLib and ARENALib implementation

The behavior of the SIMANLib components has been formally described in terms of atomic Parallel DEVS models [22]. The use of Parallel DEVS as a base to describe the behavior of SIMANLib components has facilitated the development, maintenance and reuse of the models. The formal specification of the SIMANLib components in Parallel DEVS can be found in [18]. The SIMANLib components have been developed using DEVSLib [19], a Modelica library that facilitates the description of Parallel DEVS models in Modelica. As discussed in Section 2.2, the use of DEVSLib in the implementation of SIMANLib is based on the similarities between the Parallel DEVS formalism and the Arena modeling methodology.

The SIMANLib library has been used to develop the ARENALib components in a modular fashion. ARENALib components are constructed as a combination of interconnected SIMANLib components. The same structure can be observed in the Arena environment, whose flowchart modules are constructed using the SIMAN language. The behavior of ARENALib components is formally described as coupled Parallel DEVS models, since SIMANLib components are described as atomic Parallel DEVS models.